Rechnerstrukturen

Michael Engel und Peter Marwedel

TU Dortmund, Fakultät für Informatik

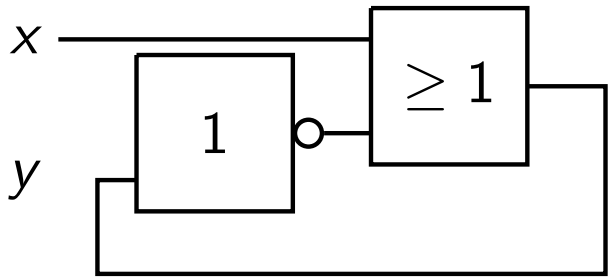
SS 2013

Hinweis: *Folien a. d. Basis von Materialien von Gernot Fink und Thomas Jansen*

13. Mai 2013

1. Sequenzielle Schaltungen
   - Einleitung (Wiederholung)
   - Modellierung mit Automaten
2. Synchrone Schaltwerke
   - Einleitung
   - Flip-Flops
3. Schaltwerk-Entwurf
   - Einleitung
   - von Neumann-Addierwerk

Eine konkrete sequenzielle Schaltung



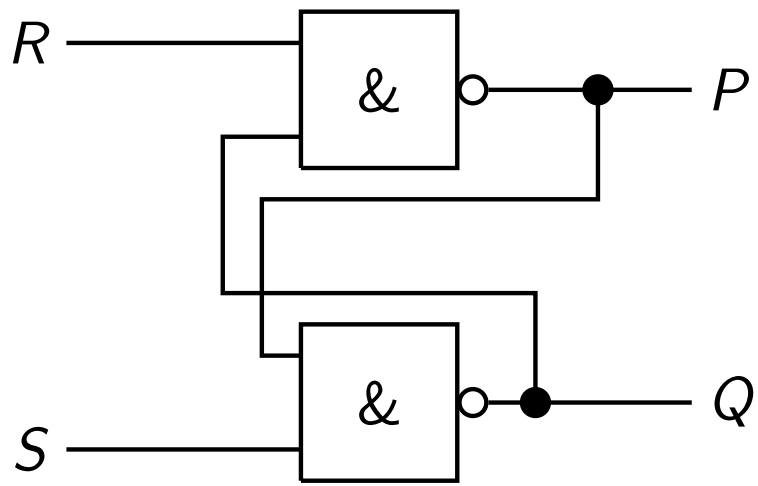
Was passiert in dieser Schaltung?

x	y	$x \vee \overline{y}$
0	0	1
0	1	0
1	0	1
1	1	1

klar und immer so weiter...

natürlich in der Realität viel schneller
darum heißt die Schaltung Flimmerschaltung

Bi-stabile NAND-Kippstufe



Anmerkung

heißt auch Latch

positiv kippt, flimmert nicht

negativ Verhalten hängt von Schaltzeiten der beiden Gatter ab

genauer beobachtet Verhalten hängt manchmal von Schaltzeiten der beiden Gatter ab

Fazit der Analyse der bi-stabilen NAND-Kippstufe

1. Fall oberes NAND-Gatter schneller
   $P_{t+2\Delta} = \overline{R} \vee S\,\overline{Q_t}$
   $Q_{t+2\Delta} = S \vee R\,Q_t$
2. Fall unteres NAND-Gatter schneller
   $P_{t+2\Delta} = \overline{R} \vee S\,P_t$
   $Q_{t+2\Delta} = S \vee R\,\overline{P_t}$

also Wenn $Q_t = \overline{P_t}$, so ist das Verhalten stabil, von den Schaltzeiten der Gatter unabhängig.

also Forderung $P_t \neq Q_t$

Wertetabelle bi-stabile NAND-Kippstufe

		oberes Gatter schneller		unteres Gatter schneller	
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	1	1	1	1
0	1	1	0	1	0
1	0	0	1	0	1
1	1	$\overline{Q_t}$	Q_t	P_t	$\overline{P_t}$

Beobachtung Wir müssen nur $R = S = 0$ ausschließen.

R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	1	1	0
1	0	0	1
1	1	P_t	$\overline{P_t}$

- $(R, S) = (0, 1)$ setzt $P = 1$
- $(R, S) = (1, 0)$ setzt $P = 0$
- $(R, S) = (1, 1)$ lässt P unverändert

Fazit Bi-stabile NAND-Kippstufe realisiert 1-Bit-Speicher!

Erstes Fazit zu sequenziellen Schaltungen

Bi-stabile NAND-Kippstufe realisiert 1-Bit-Speicher.

also

- Kreise in „Schaltnetzen“ manchmal sinnvoll
- neue Funktionalität
- Analyse schwierig

Wunsch strukturierter Entwurf

Automaten

Wunsch formales Modell eines Automaten

Was ist überhaupt ein Automat?

Beispiele

- Getränke-Automat
- einfache Ampelsteuerung
- Steuerung einer Waschmaschine

Gegenbeispiele

- Geldspielautomat
  wegen der Zufalls-Komponente
- Computer
  zu komplex
- Mensch
  für uns nicht formal beschreibbar

Automatenmodell

Grobbeschreibung

- verarbeitet eine Eingabe
- erzeugt eine Ausgabe
- ist in einem Zustand
- arbeitet in Takten
- arbeitet deterministisch (exakt vorhersagbar)

jetzt exakte, formale Beschreibung

Definition Mealy-Automat

Definiton 19

Ein Mealy-Automat $M = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$ ist definiert durch:

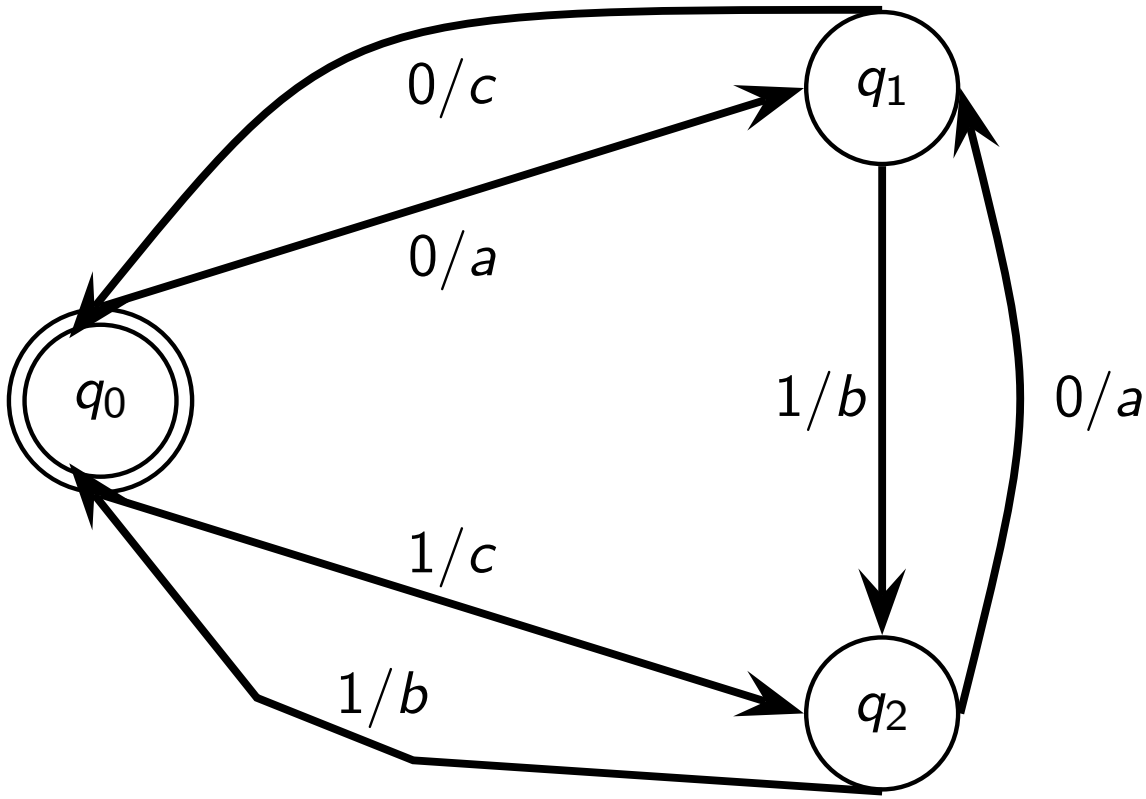
- endliche Zustandsmenge Q
- Startzustand $q_0 \in Q$
- endliches Eingabealphabet Σ
- endliches Ausgabealphabet Δ
- Zustandsüberführungsfunktion $\delta\colon Q \times \Sigma \to Q$
- Ausgabefunktion $\lambda\colon Q \times \Sigma \to \Delta \cup \{\varepsilon\}$

In einem Takt mit aktuellem Zustand q und Eingabesymbol w

- schreibt der Automat $\lambda(q, w)$,
- wechselt der Automat in den Zustand $\delta(q, w)$.

Beispiel Mealy-Automat

$Q = \{q_0, q_1, q_2\}$, $\Sigma = \{0, 1\}$, $\Delta = \{a, b, c\}$



Eingabe 0 1 0 0 Ausgabe a b a c

Äquivalenz von Automaten

Definition

Zwei Mealy-Automaten heißen äquivalent, wenn sie für jede Eingabe $w \in \Sigma^*$ die gleiche Ausgabe $a \in \Delta^*$ erzeugen.

klar Äquivalente Automaten können unterschiedlich groß sein.

klar Man wünscht sich möglichst kleine Automaten.

Anmerkung Komplexer Problemkreis,
umfasst auch effiziente Minimierung von Automaten
$\Rightarrow$ Näher i.d. Theoretischen Informatik (GTI/TIfAI)

Hier: noch ein anderes (ähnliches!) Automaten-Modell

Definition Moore-Automat

Definiton 20

Ein Moore-Automat $M = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$ ist definiert durch:

- endliche Zustandsmenge Q
- Startzustand $q_0 \in Q$
- endliches Eingabealphabet Σ
- endliches Ausgabealphabet Δ
- Zustandsüberführungsfunktion $\delta\colon Q \times \Sigma \to Q$
- Ausgabefunktion $\lambda\colon Q \to \Delta \cup \{\varepsilon\}$

In einem Takt mit aktuellem Zustand q und Eingabesymbol w

- schreibt der Automat $\lambda(\delta(q, w))$,
- wechselt der Automat in den Zustand $\delta(q, w)$.

Unterschied zum Mealy-Automaten: Ausgabe $\leftrightarrow$ Zustand

Mealy- und Moore-Automaten

Beobachtung Zu jedem Moore-Automaten gibt es einen äquivalenten Mealy-Automaten.

denn zu Moore-Automat $A = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$
ist Mealy-Automat $A' = (Q, q_0, \Sigma, \Delta, \delta, \lambda')$
mit $\lambda'(q, w) := \lambda(\delta(q, w))$
offensichtlich äquivalent

Beobachtung Zu jedem Mealy-Automaten gibt es eine äquivalenten Moore-Automaten.

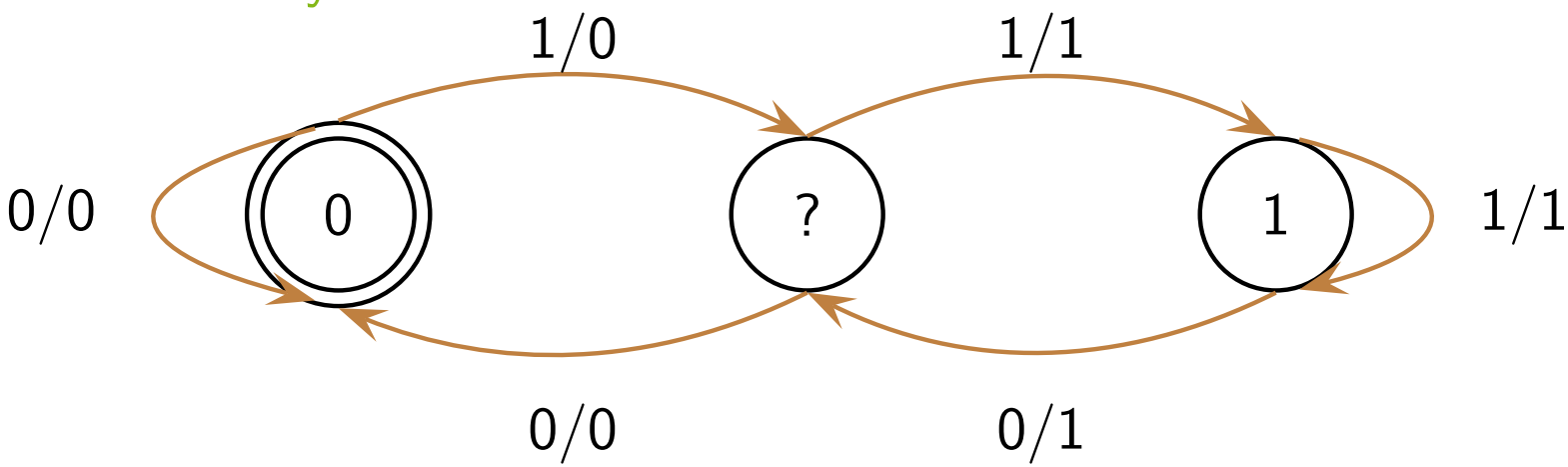
denn zu Mealy-Automaten $A = (Q, q_0, \Sigma, \Delta, \delta, \lambda)$
ist Moore-Automat $A' = (Q', q_0', \Sigma, \Delta, \delta', \lambda')$
mit $Q' := Q \times (\Delta \cup \{\varepsilon\})$, $q_0' := (q_0, \varepsilon)$,
$\delta'(q', w) = \delta'((q, v), w) := (\delta(q, w), \lambda(q, w))$,
$\lambda(\delta(q', w)) = \lambda'((q, v)) := v$
offensichtlich äquivalent

Einfacher Beispiel-Automat

Aufgabe einfache „Datenglättung“
Filtere isolierte Bits aus Datenstrom aus.

klar $\Sigma := \{0, 1\},\ \Delta := \{0, 1\}$

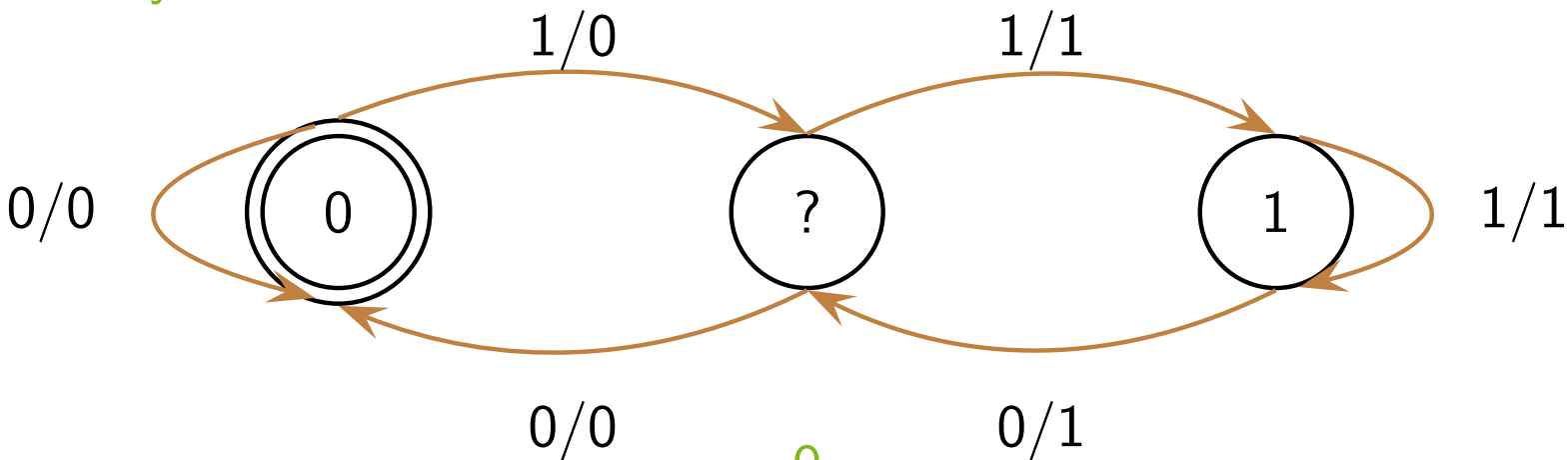
Mealy-Automat



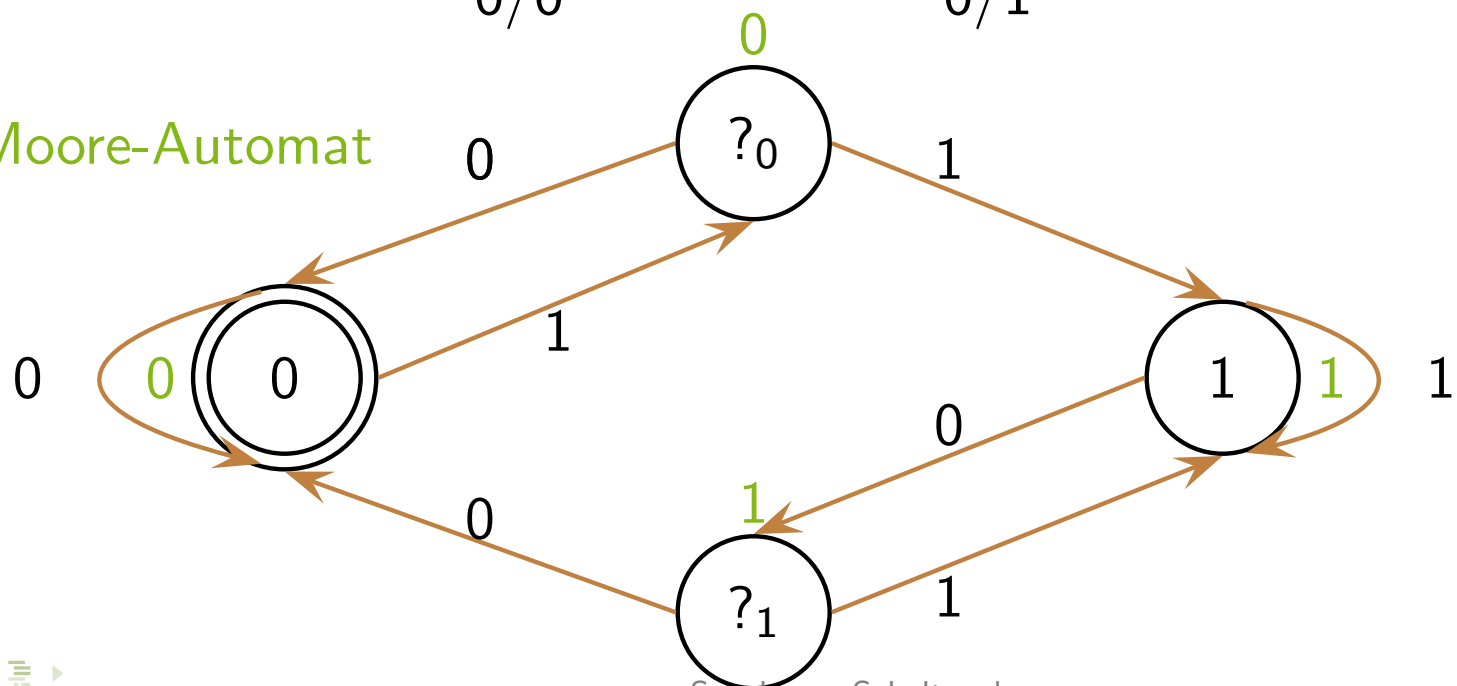
übrigens $Q := \{0, 1, ?\},\ q_0 := 0$

Äquivalente Automaten „Bit-Filter“

Mealy-Automat

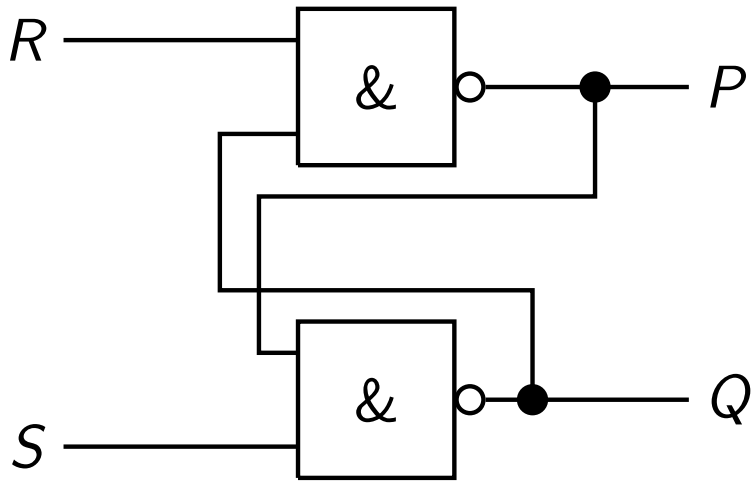


Moore-Automat

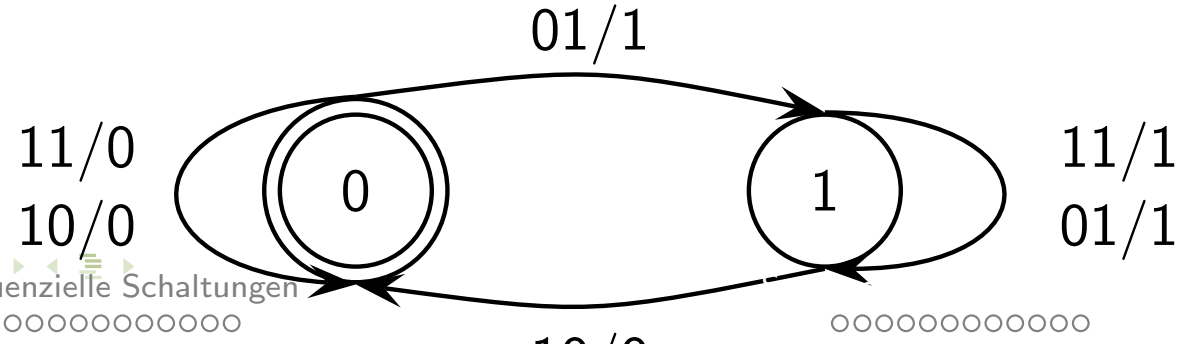


Automaten und Schaltungen: Synchrone Schaltwerke

Erinnerung bi-stabile NAND-Kippstufe



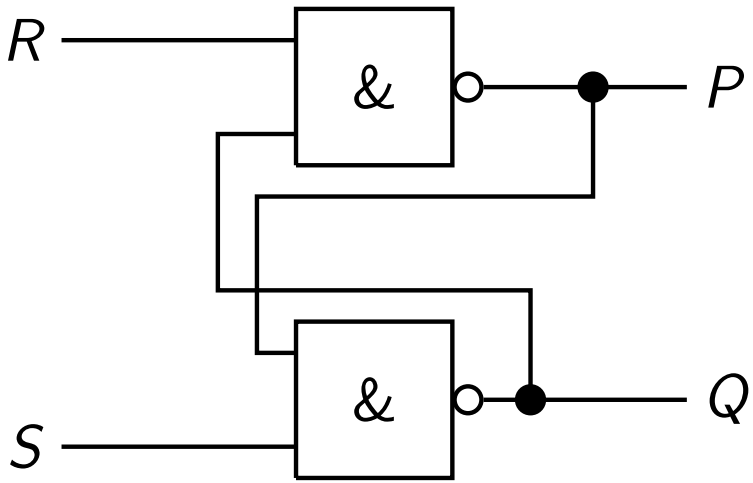
R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	1	1	0
1	0	0	1
1	1	P_t	$\overline{P_t}$



$$Q = \{0, 1\},\ q_0 = 0$$
$$\Sigma = \{01, 10, 11\}$$
$$\Delta = \{0, 1\}$$

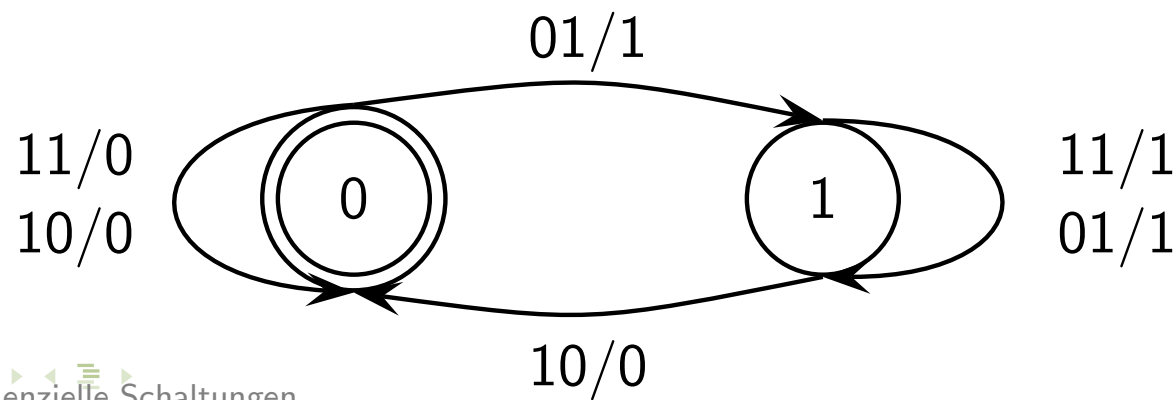
Vergleich Automat und NAND-Kippstufe

nicht getaktet



R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	1	1	0
1	0	0	1
1	1	P_t	$\overline{P_t}$

getaktet



$Q = \{0, 1\},\ q_0 = 0$
$\Sigma = \{01, 10, 11\}$
$\Delta = \{0, 1\}$

Synchrone Schaltwerke

ab jetzt getaktete Schaltwerke

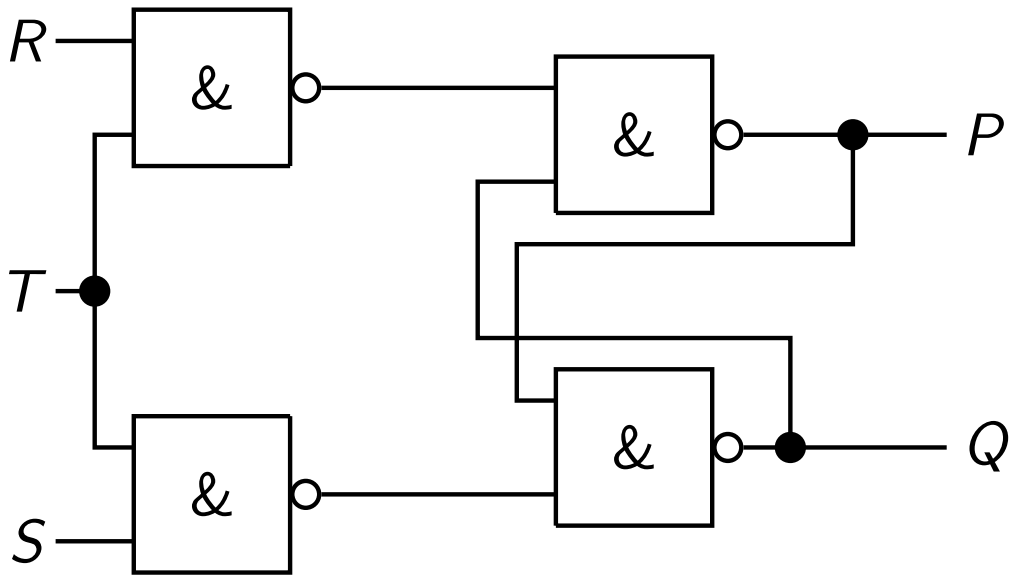
also Führe Taktsignal T ein

verschiedene technische Möglichkeiten

- Pegelsteuerung: aktiv, wenn 1 anliegt
- positive Flankensteuerung: aktiv, wenn Wechsel von 0 nach 1
- negative Flankensteuerung: aktiv, wenn Wechsel von 1 nach 0

digital-logische Ebene technisches Detail ignorieren

RS-Flip-Flop



R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

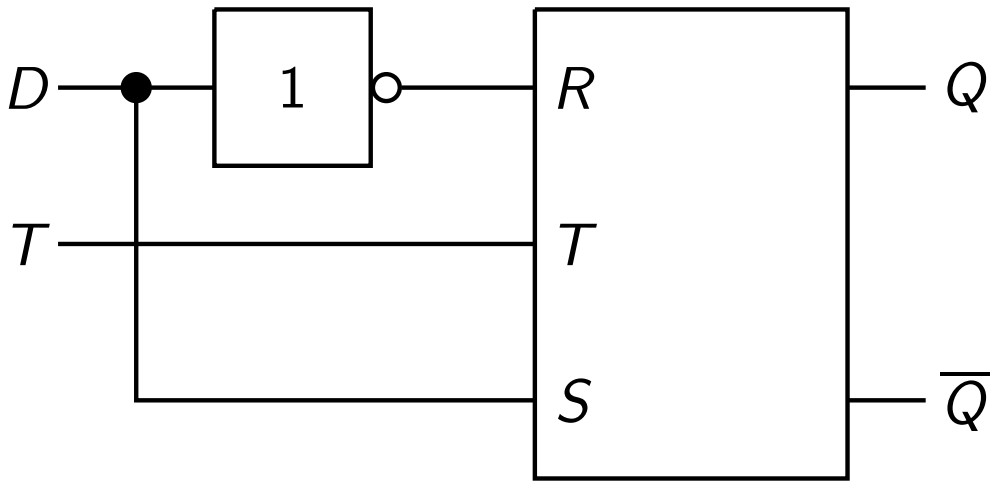
Zustandstabelle NAND-Kippstufe

R_t	S_t	$P_{t+2\Delta}$	$Q_{t+2\Delta}$
0	0	nicht erlaubt	
0	1	1	0
1	0	0	1
1	1	$\overline{Q_t}$	Q_t

Wertetabelle NAND

x	y	$\overline{x\,y}$
0	0	1
0	1	1
1	0	1
1	1	0

D-Flip-Flop



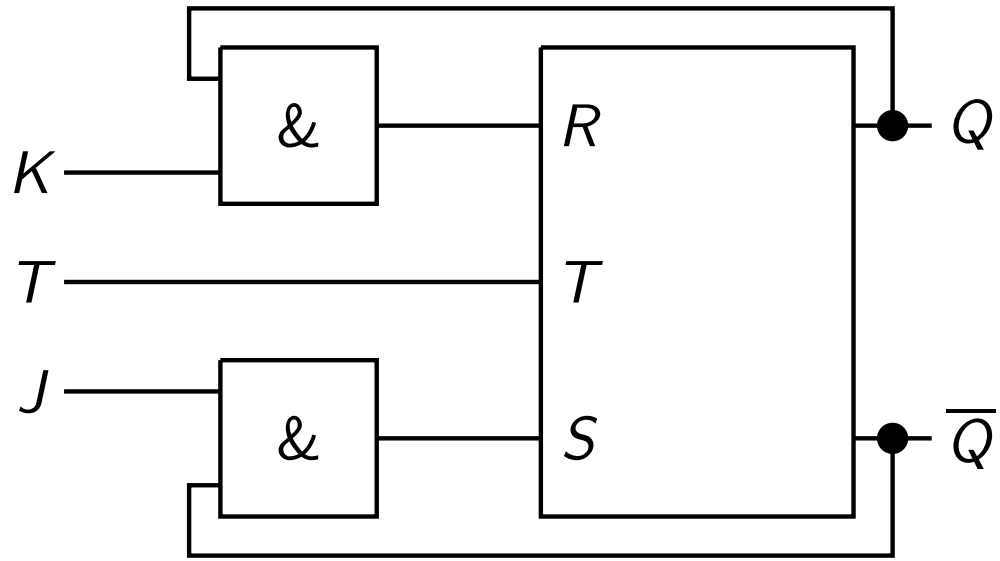
R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

D	Q
0	0
1	1

Sinnlos? Verzögerung um 1 Takt

Name Delay

JK-Flip-Flop

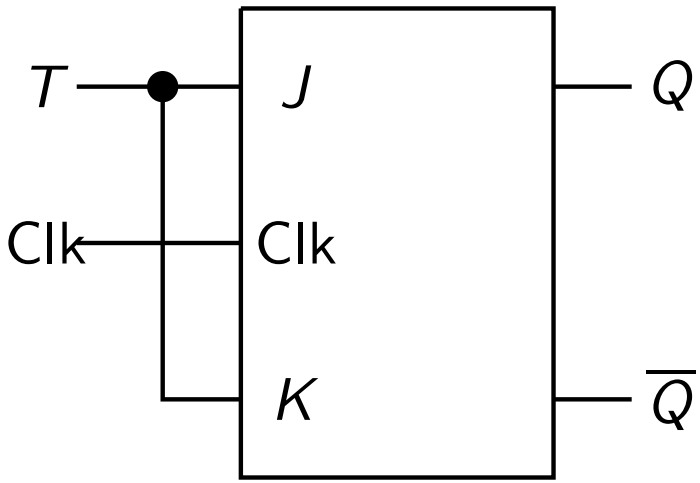


R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

J	K	R	S	Q
0	0	0	0	Q
0	1	Q	0	0
1	0	0	$\overline{Q}$	1
1	1	Q	$\overline{Q}$	$\overline{Q}$

positiv alle Eingaben erlaubt, sinnvolle Funktionalität, vielseitig

T-Flip-Flop



J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

T	Q
0	Q
1	$\overline{Q}$

Name Toggle

Flip-Flops

Wir haben also hier 4 verschiedene Flip-Flop-Typen

Wozu brauchen wir eigentlich Flip-Flops?

klar Realisierung von Speicher

Was müssen wir für den Einsatz als Speicher wissen?

klar gezielte Änderung von Speicherinhalten

also Zustandstabellen eigentlich nicht interessant

besser Ansteuertabellen

etwas präziser

Zustandstabelle			Ansteuertabelle		
Eingabe	$\Rightarrow$	Zustand	Ist-Zustand, Soll-Zustand	$\Rightarrow$	Eingabe

Anmerkung Ansteuertabellen können „don't care“ enthalten

Ansteuertabelle D-Flip-Flop

Zustandstabelle

D	Q
0	0
1	1

Ansteuertabelle

Q_{alt}	Q_{neu}	D
0	0	0
0	1	1
1	0	0
1	1	1

Beobachtung Ansteuerung D vom alten Zustand unabhängig

Ansteuertabelle T-Flip-Flop

Zustandstabelle

T	Q
0	Q
1	$\overline{Q}$

Ansteuertabelle

Q_{alt}	Q_{neu}	T
0	0	0
0	1	1
1	0	1
1	1	0

Beobachtung Kenntnis des „alten“ Zustands erforderlich

Ansteuertabelle RS-Flip-Flop

Zustandstabelle

R	S	Q
0	0	Q
0	1	1
1	0	0
1	1	nicht erlaubt

Ansteuertabelle

Q_{alt}	Q_{neu}	R	S
0	0	$*$	0
0	1	0	1
1	0	1	0
1	1	0	$*$

Beobachtung wenn Zustand nicht wechselt, gibt es Freiheit in der Ansteuerung

Ansteuertabelle JK-Flip-Flop

Zustandstabelle

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Ansteuertabelle

Q_{alt}	Q_{neu}	J	K
0	0	0	$*$
0	1	1	$*$
1	0	$*$	1
1	1	$*$	0

Beobachtung immer Freiheit in der Ansteuerung

Unvollständig definierte Ansteuerfunktionen

Wir haben — für RS-Flip-Flops und JK-Flip-Flops nur partiell definierte Ansteuerfunktionen

Ist das ungünstig?

Nein!

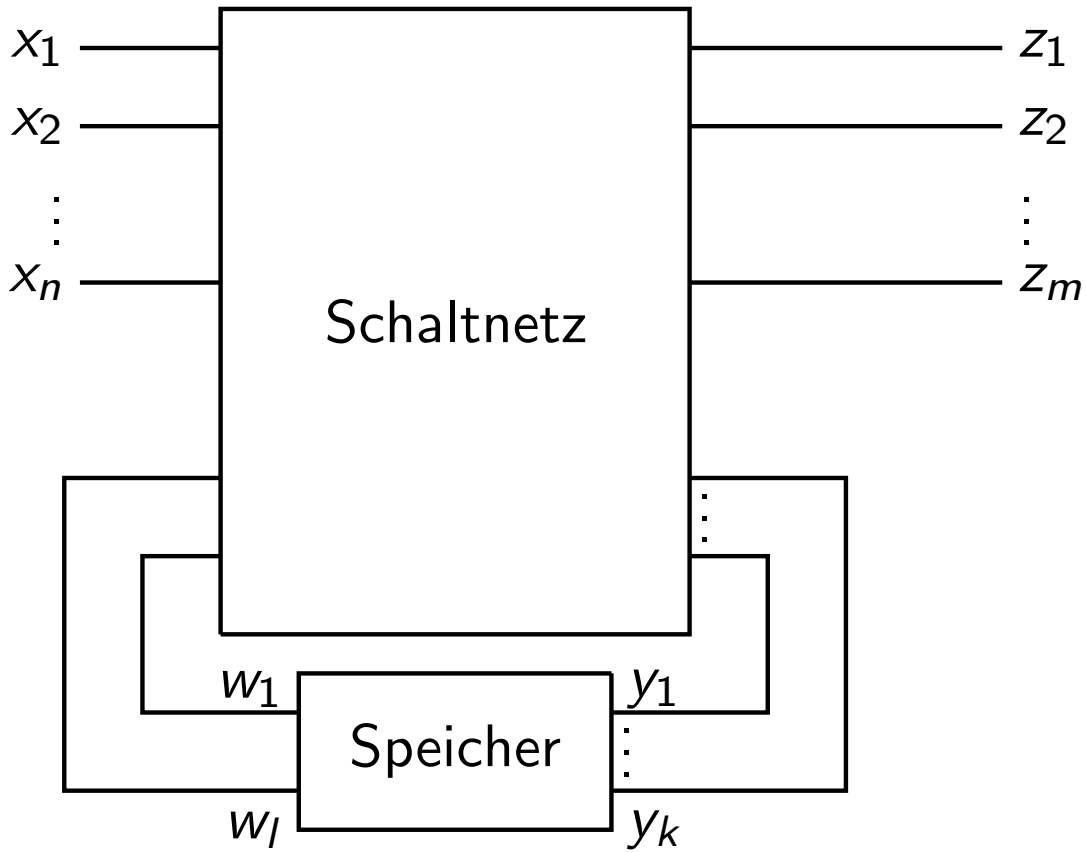
Erinnerung — Minimalpolynome für partiell definierte Funktionen

Erinnerung — Realisierungen können wesentlich einfacher sein

Erinnerung — Minimalpolynom für partiell definiertes f durch PI für f_1 und Überdeckung von f_0

Huffmann Schaltwerk-Modell

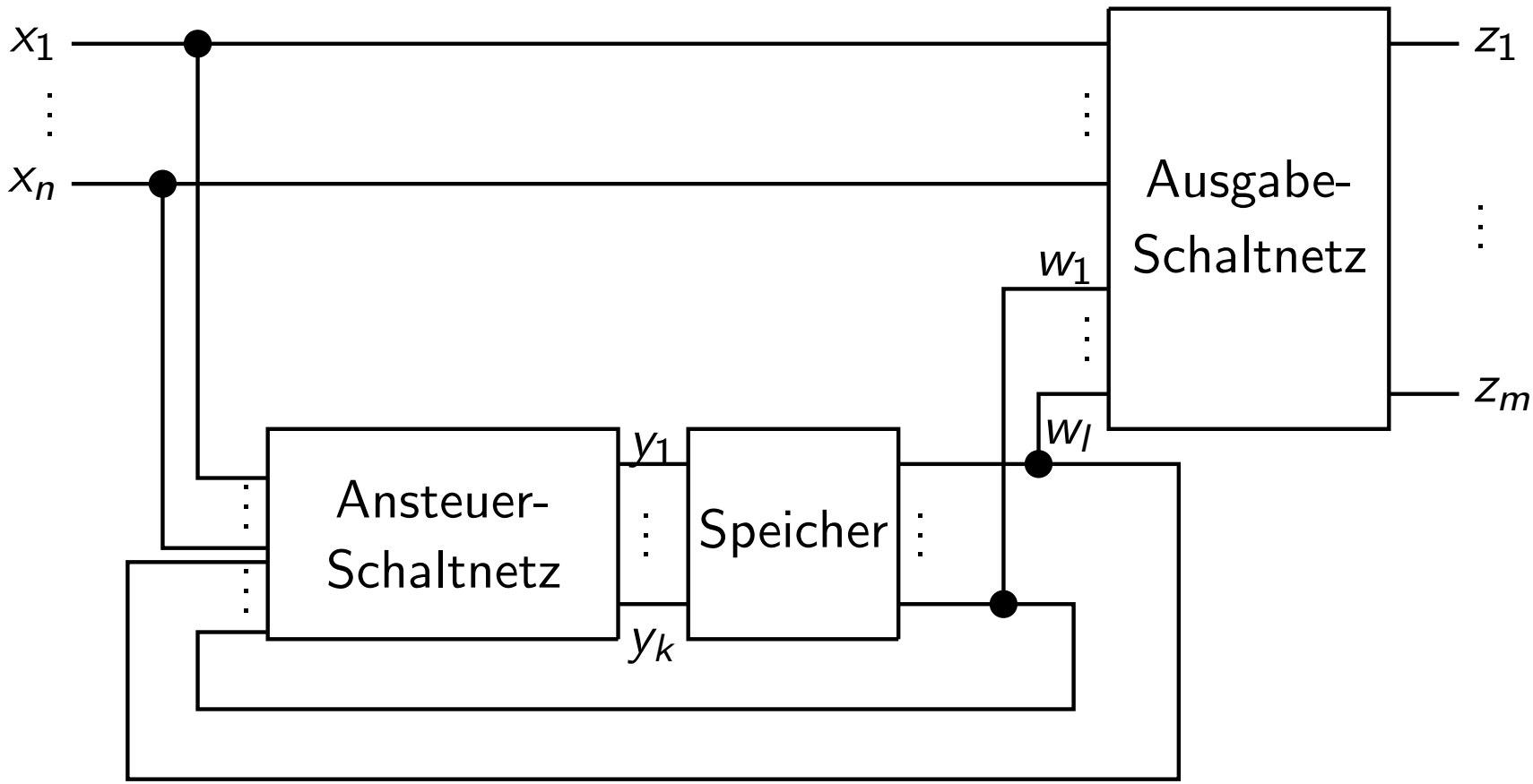
ein allgemeines, formales Schaltwerkmodell



Beobachtung führt Gelerntes über Schaltnetze, Flip-Flops und Automaten sinnvoll zusammen

Huffmann Schaltwerk-Modell

etwas detaillierter



Schaltwerk-Entwurf

Wunsch strukturierter Schaltwerk-Entwurf

Was können wir überhaupt als Schaltwerk realisieren?

klar alles, was als Automat beschrieben werden kann

zum Beispiel

- Getränke-Automat
- Ampelsteuerung
- Waschmaschinen-Steuerung
- ...

Anmerkung Heuristiken und Erfahrung sind wichtig

Schritte beim Schaltwerk-Entwurf

0. Verstehen der Aufgabe
1. Spezifikation des Verhaltens (z. B. als Mealy-Automat)
2. Wahl der Coderierung von Eingaben, Zuständen, Ausgaben
3. Wertetabelle mit Eingaben, Zustand, Ausgaben, neuem Zustand
4. Wahl der Flip-Flop-Typen
5. Ergänzung der Wertetabelle um die Flip-Flop-Ansteuerung
6. Entwurf passender boolescher Funktionen
7. Entwurf passender Schaltnetze
8. Entwurf vollständiges getaktetes Schaltwerk

Beispiel zum Schaltwerk-Entwurf

zur Einführung ein „praktisches“ Beispiel

hilfreich besonders gut verstandenes Problem

Aufgabe Addition von zwei Betragszahlen

Erinnerung wir haben

Verfahren	Größe	Tiefe
Schul-Methode	$\approx 5n$	$\approx 2n$
Carry-Look-Ahead	$\approx n^2$	$\approx 2 \log_2 n$

jetzt klein und flach mit einem Schaltwerk
aber extrem langsam

Entwurf Addierwerk: Schritt 0

Schritt 0 verstehen der Aufgabe

Wunsch Summanden bitweise eingeben
Summe bitweise erhalten

klar geht nur von rechts nach links

Was muss man sich merken?

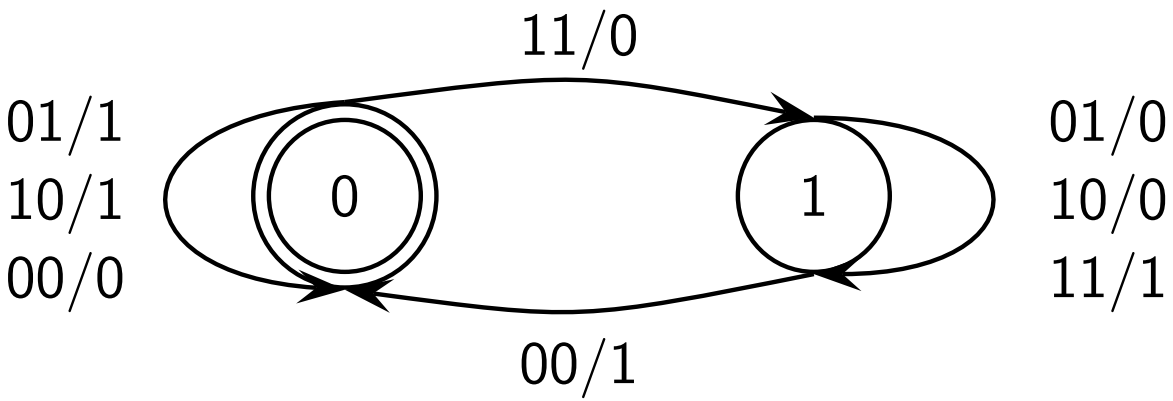
klar nur den aktuellen Übertrag
also 1 Bit

Entwurf Addierwerk: Schritt 1

Schritt 1 Spezifikation des Verhaltens

Entscheidung Beschreibung durch Mealy-Automat

Eingabealphabet $\Sigma = \{00, 01, 10, 11\}$
Ausgabealphabet $\Delta = \{0, 1\}$
Zustandsmenge $Q = \{0, 1\}$



Entwurf Addierwerk: Schritt 2

Schritt 2 Wahl der Codierung: Eingaben, Zuständen, Ausgaben

klar im Allgemeinen (fast) jede Freiheit

hier kanonische Codierung naheliegend

Codierung von Q

$q \in Q$	Codierung
0	0
1	1

Codierung von Σ

$w \in \Sigma$	Codierung
00	0 0
01	0 1
10	1 0
11	1 1

Codierung von Δ

$w \in \Delta$	Codierung
0	0
1	1

Entwurf Addierwerk: Schritt 3

Schritt 3 Wertetabelle Eingaben, Zustand, neuer Zustand, Ausgaben

naheliegend Eingaben heißen x, y
alter Zustand heißt c_{alt}
neuer Zustand heißt c_{neu}
Ausgabe heißt s

c_{alt}	x	y	c_{neu}	s
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

Entwurf Addierwerk: Schritt 4

Schritt 4 Wahl der Flip-Flop-Typen

Entscheidung JK-Flip-Flops

Anmerkung

- nicht viele überzeugende Gründe
- weil es besonders viele Freiheiten erlaubt
- weil der Zustandswechsel nichts nahelegt
- weil es oft benutzt wird

Enturf Addierwerk: Schritt 5

Schritt 5 Ergänzung der Wertetabelle um Flip-Flop-Ansteuerung

c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	$*$
0	0	1	0	1	0	$*$
0	1	0	0	1	0	$*$
0	1	1	1	0	1	$*$
1	0	0	0	1	$*$	1
1	0	1	1	0	$*$	0
1	1	0	1	0	$*$	0
1	1	1	1	1	$*$	0

Ansteuertabelle JK-Flip-Flop

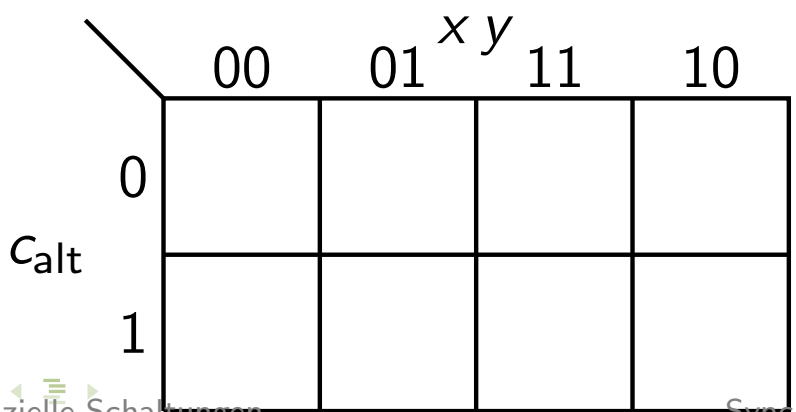
Q_{alt}	Q_{neu}	J	K
0	0	0	$*$
0	1	1	$*$
1	0	$*$	1
1	1	$*$	0

Entwurf Addierwerk: Schritt 6

Schritt 6 Entwurf passender boolescher Funktionen

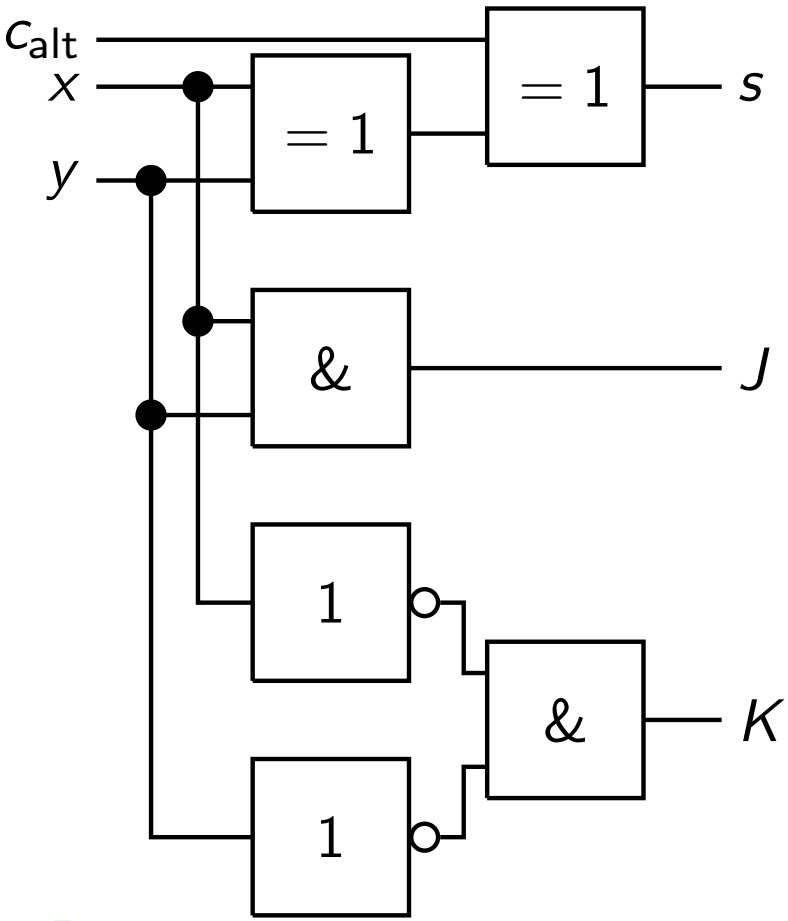
c_{alt}	x	y	c_{neu}	s	J	K
0	0	0	0	0	0	$*$
0	0	1	0	1	0	$*$
0	1	0	0	1	0	$*$
0	1	1	1	0	1	$*$
1	0	0	0	1	$*$	1
1	0	1	1	0	$*$	0
1	1	0	1	0	$*$	0
1	1	1	1	1	$*$	0

$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$
$$s(c_{alt}, x, y) = c_{alt} \oplus x \oplus y$$



Entwurf Addierwerk: Schritt 7

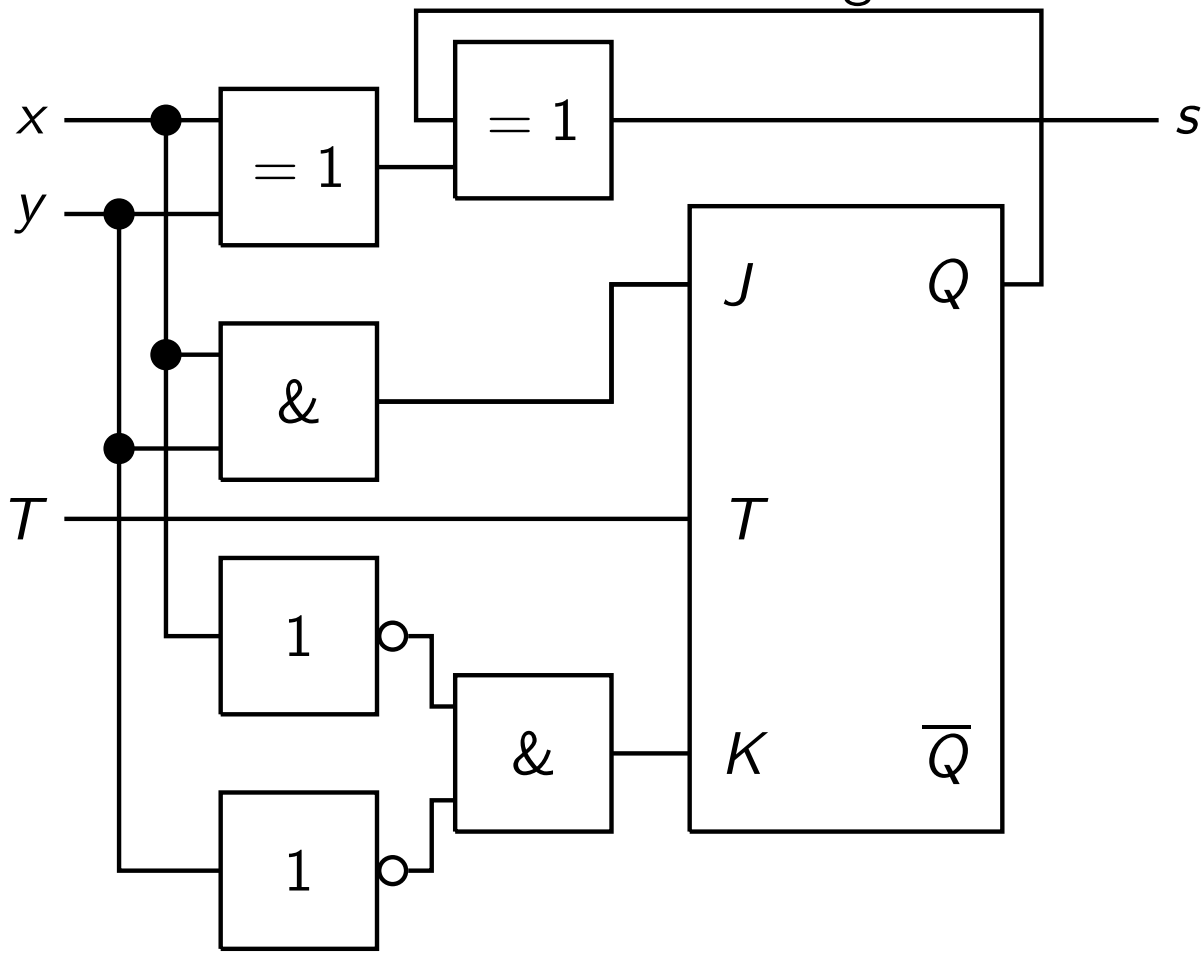
Schritt 7 Entwurf passender Schaltnetze



$$J(c_{alt}, x, y) = x\,y$$
$$K(c_{alt}, x, y) = \overline{x}\,\overline{y}$$
$$s(c_{alt}, x, y) = c_{alt} \oplus x \oplus y$$

Entwurf Addierwerk: Schritt 8

Schritt 8 Entwurf des vollständigen Schaltwerks



$$\begin{array}{r} 0\ 1\ 0\ 0\ 1\ 1 \\ +\ 0\ 1\ 0\ 1\ 0\ 1 \\ \hline 1\ 0\ 1\ 0\ 0\ 0 \end{array}$$

Beobachung Eingabe $(0, 0)$ im letzten Takt wichtig

Initialisierung? Eingabe $(0, 0)$

Unser Addierwerk: Serien-Addierer

Fazit

- addiert beliebig lange Betragszahlen
- ist sehr klein und flach
- ist sehr leicht zu initialisieren
- ist extrem langsam

Warum ist das Schaltwerk so langsam?

klar und bekannt wegen der Überträge

Müssen Überträge so lange dauern?

bekannt im Allgemeinen nicht (siehe Addierer)

aber Bei bit-weiser Eingabe schon!

Über unsere Modelle

Wir wissen schon Überträge werden nur manchmal lange weitergereicht.

Kann man ein Schaltwerk bauen, dass nur manchmal langsam ist?

Problem unsere Automaten können das zunächst nicht

Beobachtung bei Mealy- und Moore-Automat bestimmt Länge der Eingabe die Anzahl der Rechentakte

darum Modifikation des Automatenmodells

neu Erlaube leere Eingabe ε und signalisiere Ende der Rechnung durch Rechenende-Zeichen

Beobachtung Das ist fundamental neu für uns. Rechenzeit kann von der Eingabe abhängen (nicht nur von der Eingabelänge).

Auf dem Weg zum besseren Addierwerk

Wie wollen wir vorgehen?

Beobachtung Eingaben müssen sofort ganz zur Verfügung stehen
sonst kann man nicht schneller sein

also Eingabealphabet $\Sigma = \{0,1\}^{2n}$

Eingabe $x_{n-1}x_{n-2}\cdots x_1x_0y_{n-1}y_{n-2}\cdots y_1y_0 \in \{0,1\}^{2n}$

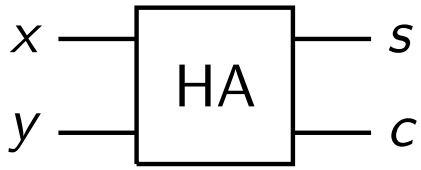
interpretieren als

$$\begin{array}{cccccc} & x_{n-1} & x_{n-2} & \cdots & x_0 \\ + & y_{n-1} & y_{n-2} & \cdots & y_0 \\ \hline \end{array}$$

bekannte Idee zur Lösung
Ersetze x und y durch x' und y' mit $x + y = x' + y'$
so lange, bis $y' = 0$ gilt.

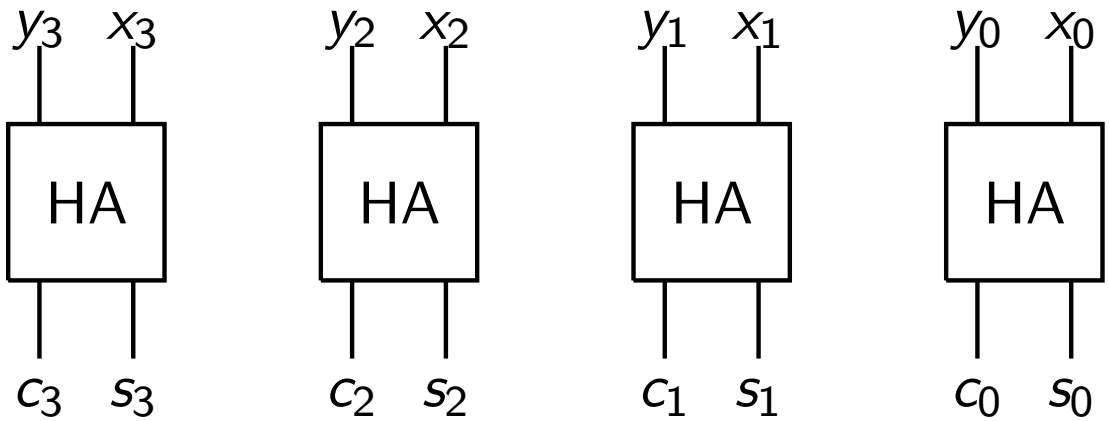
Eine gute Idee verallgemeinern

Erinnerung Wir kennen das schon vom Halbaddierer...



mit $x + y = 2c + s = (c0)_2 + s$

klar Das funktioniert auch für alle n Stellen parallel.



$$x_3' \, x_2' \, x_1' \, x_0' \quad = \quad s_3 \, s_2 \, s_1 \, s_0$$
$$\text{Ü} \; y_3' \, y_2' \, y_1' \, y_0' \quad = \quad c_3 \, c_2 \, c_1 \, c_0 \, 0$$

Fortschritt? in y hinten „neue“ 0

also nach $\leq n$ Takten $y = 0$

Noch offene Fragen

Was ist mit dem Ü?

klar potenziell kann in jedem Takt vorne ein Überlauf entstehen

Also bis zu n Überläufe speichern?

zum Glück nein

Wir wissen höchstens 1 Überlauf insgesamt

Wann ist die Rechnung fertig?

klar Rechnung fertig $\Leftrightarrow y = 0$

also „done“ $d = \overline{y_0 \vee y_1 \vee \cdots \vee y_{n-1}} = \neg \bigvee_{i=0}^{n-1} y_i$

jetzt unsere Ideen zusammensetzen